Journal of Multidisciplinary Engineering Science and Technology (JMEST)
ISSN: 2458-9403
Vol. 3 Issue 10, October - 2016

# Design and Simulation of Rate One-Third Convolutional Codes with Viterbi Algorithm based Hidden Markov Model for Digital Communications

**Do Duy Tan[a] and Yeon-Mo Yang[b*]**
[a]Electrical and Electronic Eng., Ho Chi Minh City University of Tech. and Education, Vietnam
[b]School of Electronic Eng. Kumoh National Inst. of Tech., Korea
[a]tandd@hcmute.edu.vn, [b]yangym@vivaldi.kumoh.ac.kr
*Correspondence: yangym@vivaldi.kumoh.ac.kr

***Abstract*— Convolutional codes are a type of channel coding used in numerous applications in order to achieve reliable data transfer, including digital video, radio, mobile communications and satellite communications. In this paper, we review some basic concepts of convolutional codes and their implementation in practical digital communication systems. We then describe the design of a convolutional decoder based on a Viterbi algorithm under the hidden Markov model (HMM). The paper specifically describes the main points of rate one-third convolutional code and its implementation in the encoder and decoder. We evaluate the performance of Viterbi decoding through simulation for both soft and hard-decision coding and by comparison with an uncoded theoretical case. Simulation results show that the system with convolutional coding obtains better quality Bit Error Rate (BER) than uncoded code words.**

***Keywords*— Channel coding, Coding gain, Convolutional code, Hidden Markov model, Viterbi decoding, Bit Error Rate, Soft and Hard Decision.***

## I. INTRODUCTION

Error control coding plays an important role in the protection of information delivery from a source to a destination with minimal errors. There are several general techniques for the control of errors. They are chosen based on the nature of the data and the user's requirements, as well as the complexity required for error-free reception [1]. Figure 1 represents key elements in a digital communication system where an information channel exists between source and destination.

The codes for error control are divided into two categories: block codes and trellis codes. Convolutional codes belong to the trellis codes category. Many applications in telecommunications have used convolutional codes because of their ability to deliver significant coding gains over the additive white Gaussian noise (AWGN) channel [2-4]. In this paper, we review some features of convolutional codes and an example implementation with an encoder and decoder through a Viterbi decoding algorithm based on the hidden Markov model (HMM)[16]. Furthermore, we analyze the differences between soft-decision and hard-decision decoding in the system model as compared to the uncoded theoretical case. Moreover, the paper presents the effect of the coding model on system resources [5-7].

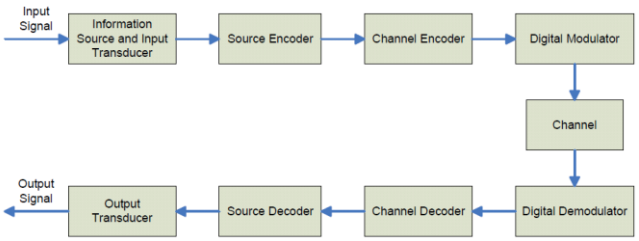


Fig. 1. *Basic key elements of a digital communication system in coding and decoding*

The remainder of this paper is organized as follows: In section II, the convolutional code concept is described through an example; parameters from this example will be used in a subsequent chapter for decoder and simulation. Section III presents a method for convolutional decoding using a Viterbi Algorithm. To estimate the performance of decoding models, some simulations for encoding and decoding are discussed in Section IV. Section V concludes with the simulation results and developments.

## II. CONVOLUTIONAL CODE IMPLEMENTATION

Convolutional codes are part of a family of codes called trellis codes. A linear trellis code is a convolutional code. We describe the encoder used for convolutional coding through a simple example. These parameters will be used in the decoder implementation in a subsequent step. Figure 2 illustrates a typical example of a rate one-third convolution encoder [1], [7-11].

### *A. Encoder*

Some notations for convolutional encoder:

- Input bits - k, the number of bits taken into the encoder at once.
- Output bits - n, the number of bits output from the encoder at once.

• Constraint length - K, the total number of shift register stages in the encoder.
• Code rate - k/n, ratio input bits and output bits.
• Generator polynomial - G(n), respectively input, relation between shift register and output.

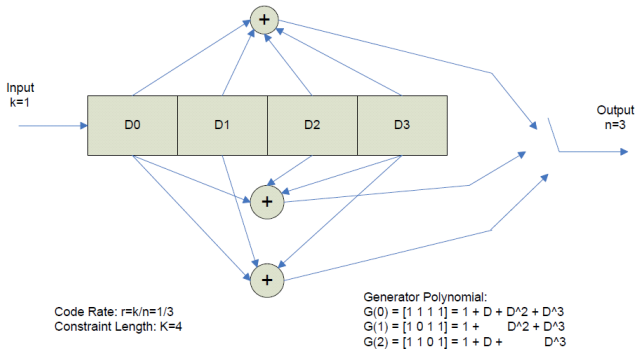


Fig. 2. *An example of a rate one-third convolutional encoder*

We chose parameters for convolutional encoder as follows:

• Input : k =1
• Output : n = 3
• Constraint length: K = 4
• Code rate: k/n = 1/3
• Generator Polynomial:

$$G(0) = [1111] = 1 + D + D^2 + D^3$$
$$G(1) = [1011] = 1 + D^2 + D^3$$
$$G(2) = [1101] = 1 + D + D^3$$

### B. *State Diagram*

The states represent the possible content of the right-most K-1 stages of the shift register (in this case K-1 is D1-D2-D3). The state diagram shows transitions from each state, corresponding to input bits. This specific example contains 8 states that come from K−1 = 3 bits. In Figure 3, we see that each input bit is 0 or 1, the state diagram changes from this value to another value, and an output code word, labeled respectively.

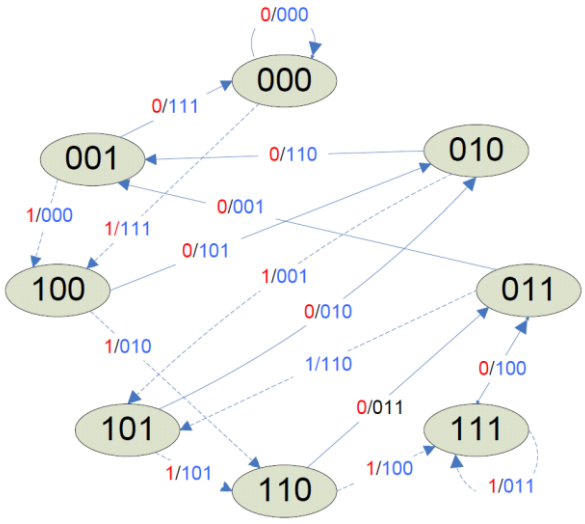


Fig. 3. *State transition diagram, labeled according to each input and output for each state*

### C. *Input/Output for Current and Next State*

Figure 4 describes the relationship between input bits and output bits through states of the shift register. The outcome is a combination of the input bit and a generator polynomial. We have a 3-bit output from a 1-bit input. Figure 4 reflects the relationship between current states and outputs consider the next state.

| Input | Current State | Next State | Output |
|---|---|---|---|
| 0 | 000 | 000 | 000 |
| 1 | 000 | 100 | 111 |
| 0 | 001 | 000 | 111 |
| 1 | 001 | 100 | 000 |
| 0 | 010 | 001 | 110 |
| 1 | 010 | 101 | 001 |
| 0 | 011 | 001 | 001 |
| 1 | 011 | 101 | 110 |
| 0 | 100 | 010 | 101 |
| 1 | 100 | 110 | 010 |
| 0 | 101 | 010 | 010 |
| 1 | 101 | 110 | 101 |
| 0 | 110 | 011 | 011 |
| 1 | 110 | 111 | 100 |
| 0 | 111 | 011 | 100 |
| 1 | 111 | 111 | 011 |

Fig. 4. *Input/output for current and next states*

### D. *Trellis Diagram*

In drawing the trellis diagram, we use the same convention that we introduced with the state diagram. The solid line denotes the output generated by an input bit, zero, and the dashed line denotes the output generated by an input bit, one. The trellis diagram contains states like the state diagram, but adds the time dimension. Each of the states can transition to one of two states. There are two outgoing branches; one corresponds to an input bit, zero, and the other corresponds to an input bit, one. Therefore, one transition from the current state to the next state will produce one output bit with one corresponding input bit.

Figure 5 shows the trellis diagram for our example through t1-t9. We refer to this time as the start time and label it as t1. The length of the trellis is referred to as the decoding window to protect a converging point for the decoding algorithm [1].

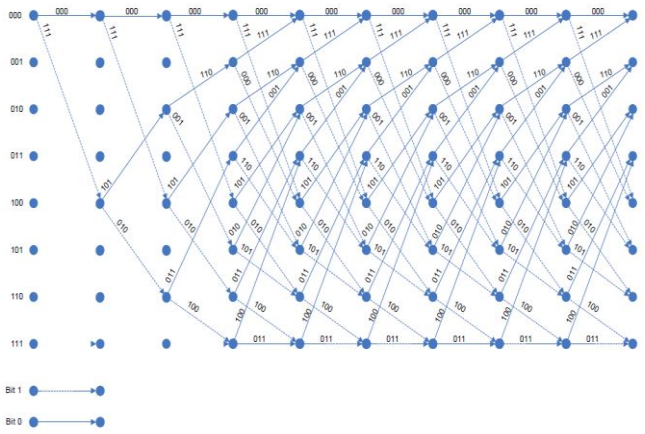

Fig. 5. *Trellis Diagram, labeled with inputs for each path*

### *E. Minimum Distance by Mason Equation*

Minimum distance or free distance can be interpreted as the minimal length of an erroneous "burst" at the output of a convolutional decoder. Figure 6 presents the computation of free distance starting with the state diagram. First, we label the branches of the state diagram with an exponent, where the exponent D denotes the Hamming distance from the branch word of that branch to the all-zeros branch. The number of exponents for each branch is the number of one bits at the output. All paths originating at a=000 and terminating at i=000 can be traced on the modified state diagram. The transfer function, T(D), is called the generating function of the code expressed as the ratio between X (at terminating transition) and X (at starting transition).

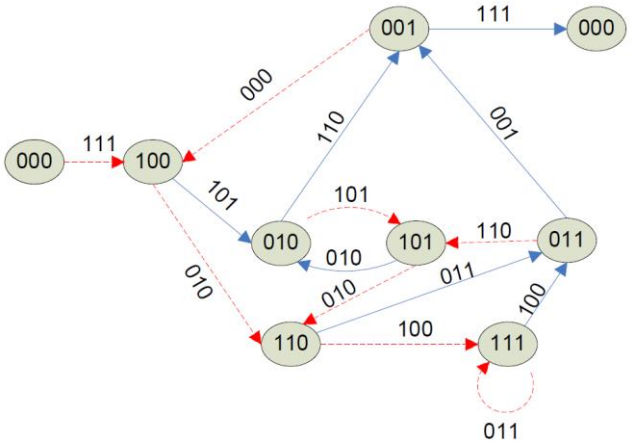


Fig. 6. *State transfer function*

Table 1 shows the state transition equation for each node. The exponent of D represents the cumulative number of ones in the path. In evaluating distance properties, the transfer function, T(D), cannot be used for long constraint lengths because the complexity of T(D) increases exponentially with constraint length.

TABLE I. STATE TRANSITION EQUATION AMONG NODES

Let: a=000, b=001, c=010, d=011, e=100, f=101, g=110, h=111, i=000. Re-write the state equation as:

$$X_e = D^3X_a + X_b$$
$$X_b = D^2X_c + DX_d$$
$$X_C = D^2X_e + DX_f$$
$$X_f = D^2X_c + D^2X_d$$
$$Xd = D^2X_g + DX_h$$
$$X_g = DX_e + DX_f$$
$$X_h = DX_g + D^2X_h$$
$$X_i = D^3X_b$$

where, $X_a$, ...,$X_i$ are dummy variable for the partial paths to the inter mediate nodes. $T(D)=X_i/X_a$ is the generating function of the code. By solving the state equation above:

$$T(D) = \frac{X_i}{X_a} = \frac{D^{10}}{1-2D+5D^2} = D^{10} + \cdots$$

Where, free distance $d_f$ of the coded is given by $d_f = 10$.

### *F. Tail-biting Method*

• Zero-tailing method: append a "tail" of M zeros (memory depth of the encoder) to the message sequence, so that at the end, the encoder memory contains only zeros and the encoder is at the all-zero state. The method is simple to implement, but due to the addition of the extra bits, the effective coding rate is reduced to l/(l+M) (where l is the length of the actual message sequence).

• Tail-biting method: initially set the encoder to a state that is identical to its final state rather than to the all-zero state. At the end of encoding, the same M bits terminate the encoder at the same initial state. The coding rate is retained. The price to pay is the increase of decoder complexity.

## III. VITERBI DECODING ALGORITHM

Convolutional decoding is the process of searching for the path that an encoder has traversed. There are three main schemes for convolutional decoding: sequential decoding, majority-logic decoding, and Viterbi decoding. Sequential decoding, as the first practical decoding technique for convolutional codes, uses the Fano algorithm (sequential decoding) and the stack algorithm. The threshold-based majority-logic decoding scheme appeared some time later. The Viterbi decoding algorithm, based on HMM is optimal in the maximum-likelihood sense, and has quickly become the most widely used convolutional decoding algorithm in practice due to its reduced computational complexity and satisfactory performance [3]. In this study, we utilize Viterbi's algorithm for convolutional decoding as the most popular in these applications. In this section, we perform convolutional coding through soft-decision/hard-decision coding then compare performance with the uncoded theoretical case.

### *A. Soft-Decision and Hard-Decision Decoding*

• For the hard-decision case, the binary phase-shift keying (BPSK) demodulator produces hard decisions at the receiver and passes them on to the decoder. (Binary Symmetric Channel - the received sequence is binary)

• For the soft-decision case, the BPSK demodulator produces soft decisions at the receiver using the log likelihood ratio. These soft outputs are 3-bit quantized and passed on to the decoder where the received sequence is a real value.

### *B. Viterbi Algorithm*

We can choose an arbitrary node (S,t) in the trellis diagram of a code and look at all of the paths going into it. We find that there is always a path that has a shorter distance between the received sequence and the code sequence than all other paths. Relying on the definition of maximum likelihood, the path with the least difference is called the local survivor path, or survivor.

• BM: Branch Metric - based on the distance between theory code word and received value depending on soft/hard decision.

• PM: Path Metric - the accumulation of the branch metric on the path from the beginning of the trellis up

to the current decoding point (Add Compare and Select).

In Figure 7 we can see that two metrics are used in the Viterbi algorithm: a branch metric, and a path metric. At state i, it may come from two previous states. Each of them has a metric to become state i, path metric at previous state, and the distance between received codeword and a theoretical codeword. At state i, the path metric is the shortest branch metric of those two paths. The survivor path is the path that has the absolute shortest path metric at state i.

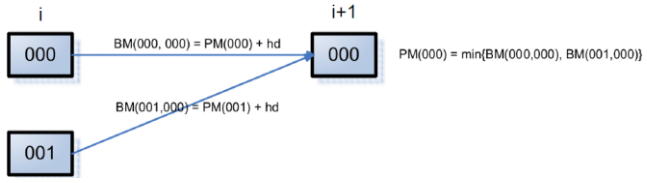


Fig. 7. *Base Metric Computation*

## IV. Experimental Results

### A. Block Diagram

Figure 8 depicts the overall configuration of the simulation environment for evaluating the proposed coding scheme [11-15]. Some parameters need to be setup for simulation, as follows: [Number of Data Bits, Traceback Window/ Queue Sizes, Shift Register, Generation Polynomial, Number of Output, Initial table for guess input from current state and previous state (based on the state diagram), Path Metric calculation (based on the state diagram), and Compare received bits after decoding with stored bits and counting errors].

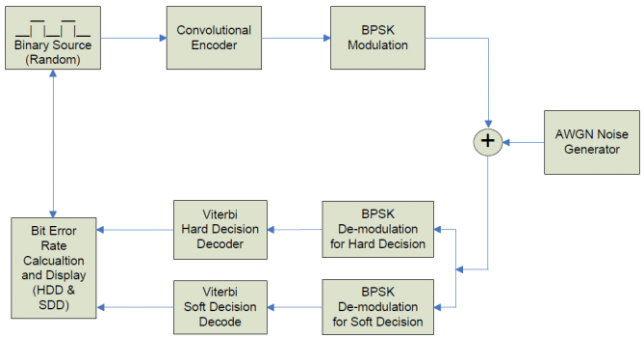


Fig. 8. *Block diagram of simulation experiments*

When using the Viterbi decoding algorithm, an important issue that needs to be taken care of is the width of the sliding window, or decoding depth. We cannot decide the outcome sequence of the decoding after only one and some state's changes because the result will be not be exact. The decoding error caused by insufficient decoding depth is called truncation error. A recommended value for this parameter should be equal to, or greater than 5 times the constraint length. In these simulations, we chose a length that is five times greater than the constraint length for the decoding depth.

The simulation starts with setting one level for Signal-to-Noise Ratio (SNR) at the transmitter. A large sequence of input bits is pushed into the encoder and BPSK modulated. The transmitted data is affected by channel noise before accessing the receiver. At the receiver side, we have two kinds of detection, one for hard-decision decoding, and the other for soft-decision decoding. The concepts of hard/soft-decision coding are mentioned in the previous section. The primary difference between them is the quantization levels, the method used to solve code distance, and complexity in the receiver. After decoding (based on decoding depth), the decoded data is stored and compared with the original sequence at transmitter to calculate bit-error rate (BER). Models are similar for C and Matlab. The flowchart for implementation of Viterbi decoding is shown in Figure 9.

1) Increase time by 1.
2) Branch Metric Computation: Compute BM for each branch.
3) Path Metric Update: Perform ACS for each current state, and store the survivor path together with its PM; discard the other(s).
4) If the end of the traceback or trellis is reached, map the global optimum path to the decode sequence and output. Otherwise go back to step 1.

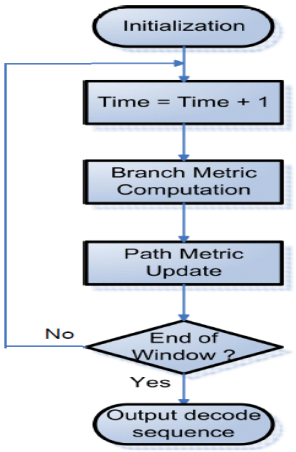


Fig. 9. *Flow chart of Viterbi decoding*

With each received bit, the algorithm calculates the branch metric and path metric for this step. After some steps, decoding depth (or traceback), a decision will be made, and the output decoded sequence is obtained. The operation continues until the end of the transmitted sequence. Similar processes repeat for other values of SNR. In these simulations, we setup 10 SNR levels and the length of sequence was 1,000,000 bits for each SNR level.

### B. Simulation in C under Cygwin

In this study, we set up two simulations (Cygwin and Matlab) to evaluate the performance of Viterbi decoding and the associated system cost. The first one, C Cygwin, is Red Hat Linux-based software, but run on Windows in a virtual environment. It helps improve computer resources and speed of simulation. In these simulations, we took into account not only the performance of the decoding algorithm, but also system resources and convergence speed for each of them when changing experiment conditions, such as decoding depth and queue size.

• Results of the simulation showed that the BER in the case of soft Viterbi decoding has a lower value than hard Viterbi decoding and the uncoded theoretical case.

• The size of the traceback window and queue size have an effect on the performance of convolutional coding and decoding. When the queue size increases, BER decreases. However, it requires more memory and simulation speed suffers. Under Cygwin, Figure 10 shows the screen shot for bit error rate(BER) probability changes versus different values of Eb/N0 when TB=20. Figure 11 plots the BER when TB=50.

```
tandd@tandd-PC /home
$ gcc BER_bpsk_Viterbi_decode_3.c

tandd@tandd-PC /home
$ ./a.exe
!BER for BPSK in AWGN with hard/soft Viterbi decoder
!g = 1111;1011;1101, TB = 20, decodeDEL = 300
!-------------------------------------------------
!Eb/N0 ! BER (sim) ! BER (sim) ! BER (theory) !
! (dB) ! (soft)    ! (hard)    ! (uncoded)    !
!-------------------------------------------------
!  0   ! 0.079779  ! 0.211019  ! 0.078650      !
!  1   ! 0.032614  ! 0.132656  ! 0.056282      !
!  2   ! 0.010219  ! 0.067775  ! 0.037506      !
!  3   ! 0.002583  ! 0.026548  ! 0.022878      !
!  4   ! 0.000605  ! 0.007977  ! 0.012501      !
!  5   ! 0.000122  ! 0.001861  ! 0.005954      !
!  6   ! 0.000023  ! 0.000309  ! 0.002388      !
!  7   ! 0.000007  ! 0.000029  ! 0.000773      !
!  8   ! 0.000002  ! 0.000005  ! 0.000191      !
!  9   ! 0.000002  ! 0.000002  ! 0.000034      !
!-------------------------------------------------
```

Fig. 10. *Screen shot under traceback = 20, queue Size = 300 in Cygwin*

```
tandd@tandd-PC /home
$ gcc BER_bpsk_Viterbi_decode_3.c

tandd@tandd-PC /home
$ ./a.exe
!BER for BPSK in AWGN with hard/soft Viterbi decoder
!g = 1111;1011;1101, TB = 50, decodeDEL = 300
!-------------------------------------------------
!Eb/N0 ! BER (sim) ! BER (sim) ! BER (theory) !
! (dB) ! (soft)    ! (hard)    ! (uncoded)    !
!-------------------------------------------------
!  0   ! 0.093376  ! 0.211069  ! 0.078650      !
!  1   ! 0.042899  ! 0.132620  ! 0.056282      !
!  2   ! 0.016484  ! 0.067753  ! 0.037506      !
!  3   ! 0.005162  ! 0.026555  ! 0.022878      !
!  4   ! 0.001320  ! 0.007974  ! 0.012501      !
!  5   ! 0.000366  ! 0.001862  ! 0.005954      !
!  6   ! 0.000054  ! 0.000310  ! 0.002388      !
!  7   ! 0.000008  ! 0.000029  ! 0.000773      !
!  8   ! 0.000002  ! 0.000005  ! 0.000191      !
!  9   ! 0.000001  ! 0.000001  ! 0.000034      !
!-------------------------------------------------
```

Fig. 11. *Screen shot under traceback = 50, queue size = 300 in Cygwin*

### C. Simulation in Matlab

The second simulation is implemented with Matlab. Targets of this simulation are similar to C, but also take care of comparison with outcomes from C. The fact is that even though it is really versatile and flexible in designing and modifying the coding algorithm under Matlab, the simulations require much greater system resources and the simulation times are longer.

• Obtained the result after 45 minutes (10 times longer than simulation by C) for case TB = 20, Queue Size = 50.

• BER also decreases when SNR increases; BER in using soft-decision decoding is smallest when compared with hard-decision decoding and theoretical uncoded communication.

• In the case of larger TB (TB=50) and Queue Size, time increases and error probability decreases. It takes more time for the simulation in Matlab than C with Cygwin.

Under Matlab, Figure 12 and 13 show BER probability changes versus different values of Eb/N0 for hard and soft Viterbi decoding different traceback size.

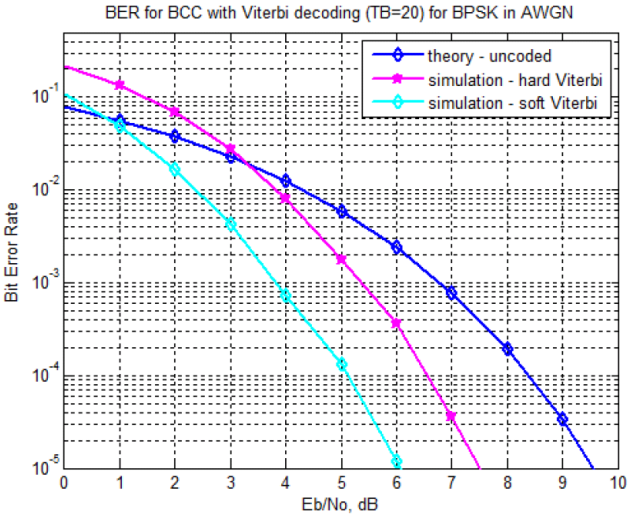


Fig. 12. *Performance evaluation for hard and soft Viterbi decoding under traceback = 20, queue size = 100*

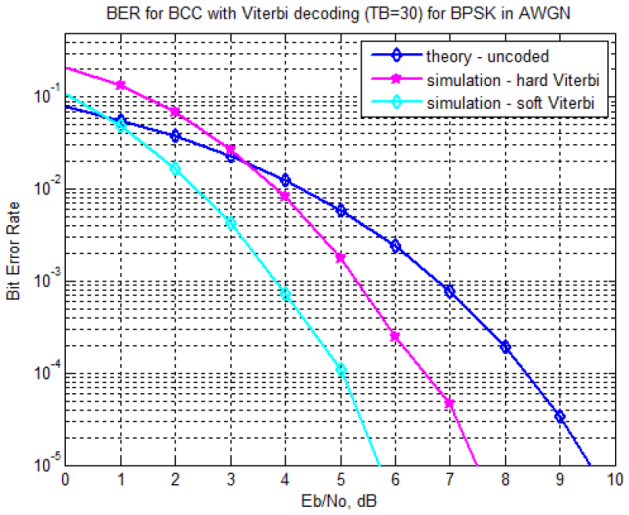


Fig. 13. *Performance evaluation for hard and soft Viterbi decoding under traceback = 30, queue size = 100*

## V. Conclusion and Further Studies

Error control coding increases performance of the communication channel at the receiver. Bit Error Rate (BER) is much better than in the case of unused error control coding because error control coding helps the receiver recognize and decide more precisely what the transmitted sequence is at the transmitter. Moreover, it

can also fix some errors based on the algorithms and characteristics of a generator polynomial.

In this paper, we present a performance evaluation for rate one-third convolutional code with a sample encoder/decoder using a Viterbi decoding algorithm based on the hidden Markov model (HMM). The paper describes the main features of the convolutional coding concept and how to implement it in a decoder using a Viterbi decoding algorithm. The results show that a significantly higher performance for BER calculation is attained. However, it requires additional complexity in both the transmitter and receiver to implement this algorithm. Moreover, the coding performance depends on the type of quantization, and whether hard-decision, or soft-decision decoding is used. These models have a significant effect on the results of coding and decoding.

In order to increase the performance of digital communication systems, we can utilize an encoder with longer constraint lengths, but with higher complexity. In addition, modulation techniques other than BPSK modulation/demodulation (e.g., Quadrature Phase Shift Keying (QPSK), Quadrature Amplitude Modulation (QAM), and Trellis Coded Modulation (TCM)) could be deployed to study the effects on performance and system resource requirements.

## X. Acknowledgment

“This paper was supported by Research Fund, Kumoh National Institute of Technology.”

## References

[1] B. Sklar, Digital Communications: Fundamentals & Applications, 2nd ed., Prentice Hall, 2001.

[2] Sweeney P. Error control coding: from theory to practice, Wiley, 2002.

[3] Yuan Jing, A Practica [4] A. El-Sherif and K. Liu, “Cooperation in random access networks: Protocol design and performance analysis,” IEEE Journal on Selected Areas in Communications, vol. 30, no. 9, pp. 1694 –1702, October 2012

[4] Soreng B. and Kumar S., "Efficient implementation of Convolution Encoder and Viterbi Decoder" ICCPCT, 2013

[5] Bin Khalid, F., Masud, S. and Uppal, M., "Design and implementation of an ML decoder for tail-biting convolutional codes," IEEE Symposium on ISCAS, 2013

[6] Junil C., Chance, Z., Love, D.J. and Madhow, U, "Noncoherent Trellis Coded Quantization: A Practical Limited Feedback Technique for Massive MIMO Systems," IEEE Transactions on Communications, Vol. 61, (12), 2013

[7] F. Dewanta and Y.-M. Yang, "Experimental Study on Performance Analysis of Viterbi Algorithm based on Hidden Markov Model considering Soft Decision Decoding," IJAER, Vol. 11, (17), pp 9185-9190, 2016

[8] Shuang F., Jie Y and Kwon, H.M, "Blind Relay Network with Viterbi Detection," IEEE MILCOM, 2014

[9] Kun Han and Deliang Wang, "Neural Network Based Pitch Tracking in Very Noisy Speech" IEEE/ACM Transactions on Audio, Speech, and Language Processing, Vol. 22, (12), 2014,

[10] Dongdong L. and Jun Y., "Efficient implementation of the decoder for tail biting convolutional codes" IEEE ICSPCC, 2014

[11] Ramteke, S., Kakde, S., Suryawanshi, Y. and Meshram, M., "Performance analysis of Turbo decoder using Soft Output Viterbi Algorithm," IEEE Communications and Signal Processing, 2015

[12] Muhammad Hassan Masood, "Convolutional Encoder and Hard Decision Viterbi Decoder," Matlab Central, 2015

[13] Krishna Sankar, "Bit Error Rate (BER) for BPSK modulation," http://www.dsplog.com, 2016

[14] Ingle and Proakis, Digital Signal Processing Using MATLAB, 4th Ed., Cengage Learning, 2017

[15] Harry Perros, Computer Simulation Techniques-The Definitive Introduction, http://www4.ncsu.edu/~hp, NCSU, 2016

[16] E. Fosler-Lussier, "Markov Models and Hidden Markov Models A Brief Tutorial," International Computer Science Institute, UC Berkeley, 1998